Esame di Fondamenti di Informatica L-B
Ingegneria Gestionale e dei Processi Gestionali (L-Z)

Appello del 21/6/2007

Compito A

Esercizio 1 (4 punti)

Visibilità di variabili, metodi e classi in Java.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
int f(int V[], int M) {
  int i=0, sum=0;
  while(++i<M)
    sum+=V[i];
  return sum;
}

int g(int V[], int N) {
  int j, tot=0;
  for(j=0; j<N/2; j++) tot+=f(V, j);
}
```

1. Calcolare la complessità in passi base della funzione `f` nei termini del parametro `M`.
2. Calcolare la complessità in passi base della funzione `g` nei termini del parametro `N` (suggerimento: si supponga `N` pari).
3. Calcolare la complessità asintotica della funzione `g` nei termini del parametro `N`.

Esercizio 3 (8 punti)

Il nuovo cinema multisala "Sergio Leone" vuole informatizzare la gestione delle prenotazioni dei posti durante le proiezioni. A tal scopo, per ogni posto presente in una sala del cinema vanno memorizzati il numero di posto, il numero della fila, e, nel caso il posto sia prenotato, il cognome del cliente che ha effettuato la prenotazione.
Si scriva una classe `Posto` per il cinema "Sergio Leone" che:

1. Possieda un opportuno costruttore con parametri (inizialmente il posto non è prenotato).
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza della classe.
3. Possieda un metodo `prenota` che, dato il cognome di un cliente, provveda a prenotare il posto, rendendolo indisponibile per altre prenotazioni.
4. Presenti il metodo `toString` che fornisca una descrizione testuale del posto (incluso il cognome del cliente nel caso il posto sia prenotato).
5. Possieda un metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Posto` (l'uguaglianza va verificata esclusivamente sul numero e sulla fila).
6. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un altro oggetto `Posto`, effettuando il confronto prima sulla fila e quindi sul numero del posto.

Esercizio 4 (8 punti)

Si scriva una classe `Sala` che memorizzi, oltre al numero della sala ed al nome del film proiettato, le informazioni relative a tutti i posti della sala all'interno di un insieme. La classe `Sala` deve:

1. Possedere un opportuno costruttore con parametri (l'insieme dei posti inizialmente è vuoto).
2. Presentare opportuni metodi che permettano di accedere alle variabili di istanza della classe.
3. Possedere un metodo `liberi` che restituisca il numero di posti liberi all'interno della sala.
4. Possedere il metodo `piena` che indichi se tutti i posti della sala sono prenotati.
5. Presentare un metodo `prenota` che, dati il cognome di un cliente, il numero di una fila e i due numeri di posto iniziale e finale, provveda a prenotare per il cliente tutti i posti della fila indicata compresi tra i due estremi, se tali posti sono tutti non prenotati.
6. Possedere il metodo `toString` che restituisca una descrizione della sala, comprese le informazioni sui posti all'interno della sala stessa.

Esercizio 5 (6 punti)

Si scriva un'applicazione per il cinema "Sergio Leone" che:

1. Crei una lista di oggetti di tipo `Sala`.
2. Lette da tastiera le informazioni relative a cinque nuove sale, provveda ad inserire tali oggetti nella lista.
3. Letto da tastiera il nome di un film, stampi su video il numero di tutte le sale non piene che proiettano tale film.
4. Per ogni sala, stampi su video il numero di posti liberi accanto al numero della sala.

Per la lettura di dati da tastiera è possibile utilizzare l'oggetto `Lettore.in`, definito all'interno del package `fiji.io`, che possiede i seguenti metodi:

- `boolean leggiBoolean()` Legge un boolean (delimitato da spazi).
- `char leggiChar()` Legge un singolo carattere.
- `double leggiDouble()` Legge un numero razionale (delimitato da spazi).
- `float leggiFloat()` Legge un numero razionale (delimitato da spazi).
- `int leggiInt()` Legge un intero (delimitato da spazi).
- `String leggiLinea()` Legge una linea di testo.
- `String leggiString()` Legge una parola senza spazi al suo interno.

Soluzione Esercizio 2

Domanda 1:

2 assegnamenti	2
`while(++i<M)`	`M`
assegnamento	`M` – 1
Totale	2 `M` + 1

Domanda 2:

2 assegnamenti	2
`for`	`N`/2 + 1
incremento di `j`	`N`/2
chiamata di `f`	`N`/2
complessità di `f`	$N^2/4$
Totale	$N^2/4 + 3N/2 + 3$

Domanda 3:
Complessità asintotica: $O(N^2)$

Soluzione Esercizio 3

```
class Posto implements Comparable<Posto> {
  private String cognome;
  private int numero, fila;
  private boolean prenotato;

  public Posto(int numero, int fila) {
    this.cognome=null;
    this.numero=numero;
    this.fila=fila;
    this.prenotato=false;
  }

  public String getCognome() { return cognome; }
  public int getNumero() { return numero; }
  public int getFila() { return fila; }
  public boolean isPrenotato() { return prenotato; }

  public void prenota(String cognome) {
    this.cognome=cognome;
    this.prenotato=true;
  }

  public String toString() {
    String s=""+fila+"."+numero;
    if(prenotato) s+=" ("+cognome+")";
    return s;
  }

  public boolean equals(Posto p) {
    return ((fila==p.fila)&&(numero==p.numero));
  }

  public int compareTo(Posto p) {
    int ret=(fila-p.fila);
    if(ret==0) ret=(numero-p.numero);
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Sala {
  private int numero;
  private String film;
  private Set<Posto> s;

  public Sala(int numero, String film) {
    this.numero=numero;
    this.film=film;
    this.s=new HashSet<Posto>();
  }

  public int getNumero() { return numero; }
  public String getFilm() { return film; }

  public int liberi() {
    int n=0;
    for(Posto p:s) if(!p.isPrenotato()) n++;
    return n;
  }

  public boolean piena() { return (liberi()==0); }

  public boolean prenota(String cognome, int fila, int da, int a) {
    for(Posto p:s)
      if((p.getFila()==fila)&&(p.getNumero()>=da)
        &&(p.getNumero()<=a)&&(p.isPrenotato())) return false;
    for(Posto p:s)
      if((p.getFila()==fila)&&(p.getNumero()>=da)&&(p.getNumero()<=a))
        p.prenota(cognome);
    return true;
  }

  public String toString() { return ""+numero+": "+film+"\n"+s; }
}
```

Soluzione Esercizio 5

```
import fiji.io.*;
import java.util.*;
class Applicazione {
  public static void main(String[] args) {
    List<Sala> s=new LinkedList<Sala>(); // domanda 1

    for(int i=0; i<5; i++) // domanda 2
      s.add(new Sala(Lettore.in.leggiInt(), Lettore.in.leggiLinea()));
    String film=Lettore.in.leggiLinea();
    for(Sala sala:s)
      if((!sala.piena())&&(sala.getFilm().equals(film)))
        System.out.println(sala.getNumero()); // domanda 3
    for(Sala sala:s) // domanda 4
      System.out.println(""+sala.getNumero()+" "+sala.liberi());
  }
}
```

Esame di Fondamenti di Informatica L-B
Ingegneria Gestionale e dei Processi Gestionali (L-Z)

Appello del 21/6/2007

Compito B

Esercizio 1 (4 punti)

Visibilità di variabili, metodi e classi in Java.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
int f(int V[], int M) {
  int sum=0;
  for(int i=0; ++i<M; ) sum+=V[i];
  return sum;
}

int g(int V[], int N) {
  int j=0, tot=0;
  while(j<N/2) tot+=f(V, j++);
}
```

1. Calcolare la complessità in passi base della funzione `f` nei termini del parametro `M`.
2. Calcolare la complessità in passi base della funzione `g` nei termini del parametro `N` (suggerimento: si supponga `N` pari).
3. Calcolare la complessità asintotica della funzione `g` nei termini del parametro `N`.

Esercizio 3 (7 punti)

Il centro di calcolo della Facoltà di Ingegneria della capitale dello Stato caraibico di St. Marquez vuole informatizzare la gestione delle prenotazioni dei propri laboratori informatici. A tal scopo, per ogni postazione di lavoro presente in un laboratorio va memorizzato, nel caso il posto sia occupato, il cognome dello studente che sta occupando la postazione.
Si scriva una classe `Postazione` per il centro di calcolo che:

1. Possieda un opportuno costruttore con parametri (inizialmente la postazione è libera).
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza della classe.
3. Possieda un metodo `occupa` che, dato il cognome di uno studente, provveda ad occupare la postazione, rendendola indisponibile per altri studenti.
4. Presenti il metodo `toString` che fornisca una descrizione testuale della postazione (incluso il cognome dello studente nel caso la postazione sia occupata).
5. Possieda un metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Postazione` (l'uguaglianza va verificata sul cognome dello studente nel caso che questa sia occupata).
6. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un altro oggetto `Postazione`, dando la precedenza alle postazioni non occupate.

Esercizio 4 (8 punti)

Si scriva una classe `Laboratorio` che memorizzi, oltre al nome del laboratorio, le informazioni relative a tutte le postazioni di lavoro all'interno di una lista. La classe `Laboratorio` deve:

1. Possedere un opportuno costruttore con parametri; inizialmente, la lista va riempita con alcuni oggetti di tipo `Postazione` (tale numero viene passato come parametro al costruttore).
2. Presentare opportuni metodi che permettano di accedere alle variabili di istanza della classe.
3. Possedere un metodo `occupate` che restituisca il numero di postazioni occupate all'interno del laboratorio.
4. Possedere il metodo `cercaPostazione` che restituisca, nel caso esista, il primo oggetto di tipo `Postazione` libero per cui le due postazioni vicine siano anch'esse libere.
5. Presentare un metodo `occupa` che, dati il cognome di uno studente ed una `Postazione`, provveda ad occupare tale postazione per lo studente indicato.
6. Possedere il metodo `toString` che restituisca una descrizione del laboratorio, comprese le informazioni sulle postazioni all'interno del laboratorio stesso.

Esercizio 5 (7 punti)

Si scriva un'applicazione per il per il centro di calcolo che:

1. Crei un insieme di oggetti di tipo `Laboratorio`.
2. Lette da tastiera le informazioni relative a tre nuovi laboratori, provveda ad inserire tali oggetti nell'insieme.
3. Per ogni laboratorio, stampi il numero di postazioni occupate accanto al nome del laboratorio.
4. Letto da tastiera il nome di uno studente, occupi per tale studente una postazione all'interno del primo laboratorio disponibile all'interno dell'insieme di cui al punto 1.

Per la lettura di dati da tastiera è possibile utilizzare l'oggetto `Lettore.in`, definito all'interno del package `fiji.io`, che possiede i seguenti metodi:

- `char leggiChar()` Legge un singolo carattere.
- `double leggiDouble()` Legge un numero razionale (delimitato da spazi).
- `float leggiFloat()` Legge un numero razionale (delimitato da spazi).
- `int leggiInt()` Legge un intero (delimitato da spazi).
- `String leggiLinea()` Legge una linea di testo.
- `String leggiString()` Legge una parola senza spazi al suo interno.

Soluzione Esercizio 2

Domanda 1:

2 assegnamenti	2
`for`	`M`
assegnamento	`M` – 1
Totale	2 `M` + 1

Domanda 2:

2 assegnamenti	2
`while(j<N/2)`	`N`/2 + 1
`tot+=f(V, j++)`	`N`/2
complessità di `f`	$N^2/4$
Totale	$N^2/4$ + `N` + 3

Domanda 3:
Complessità asintotica: $O(N^2)$

Soluzione Esercizio 3

```
class Postazione implements Comparable<Postazione> {
  private String cognome;
  private boolean occupata;

  public Postazione() {
    this.cognome=null;
    this.occupata=false;
  }

  public String getCognome() { return cognome; }
  public boolean isOccupata() { return occupata; }

  public void occupa(String cognome) {
    this.cognome=cognome;
    this.occupata=true;
  }

  public String toString() {
    if(occupata) return cognome;
    else return "libera";
  }

  public boolean equals(Postazione p) {
    if(((occupata)&&(p.occupata)&&(cognome.equals(p.cognome)))
      ||((!occupata)&&(!p.occupata))) return true;
    return false;
  }

  public int compareTo(Postazione p) {
    if(!occupata)
      if(p.occupata) return -1;
      else return 0;
    else if(p.occupata) return 0;
      else return 1;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Laboratorio {
  private String nome;
  private List<Postazione> l;

  public Laboratorio(String nome, int n) {
    this.nome=nome;
    this.l=new ArrayList<Postazione>();
    for(int i=0; i<n; i++) l.add(new Postazione());
  }

  public String getNome() { return nome; }

  public int occupate() {
    int n=0;
    for(Postazione p:l) if(p.isOccupata()) n++;
    return n;
  }

  public Postazione cercaPostazione() {
    for(int i=1; i<l.size()-1; i++)
      if((!l.get(i-1).isOccupata())&&(!l.get(i).isOccupata())
        &&(!l.get(i+1).isOccupata())) return l.get(i);
    return null;
  }

  public void occupa(String cognome, Postazione p) { p.occupa(cognome);}
  public String toString() { return nome+"\n"+l.toString(); }
}
```

Soluzione Esercizio 5

```
import fiji.io.*;
import java.util.*;
class Applicazione {
  public static void main(String[] args) {
    Set<Laboratorio> s=new TreeSet<Laboratorio>(); // domanda 1

    for(int i=0; i<3; i++)
      s.add(new Laboratorio(Lettore.in.leggiLinea(),
        Lettore.in.leggiInt())); // domanda 2
    for(Laboratorio l:s) // domanda 3
      System.out.println(l.getNome()+" "+l.occupate());
    String nome=Lettore.in.leggiLinea();
    for(Laboratorio l:s) {
      Postazione p=l.cercaPostazione();
      if(p!=null) {
        l.occupa(nome, p);
        break;
      }
    } // domanda 4
  }
}
```